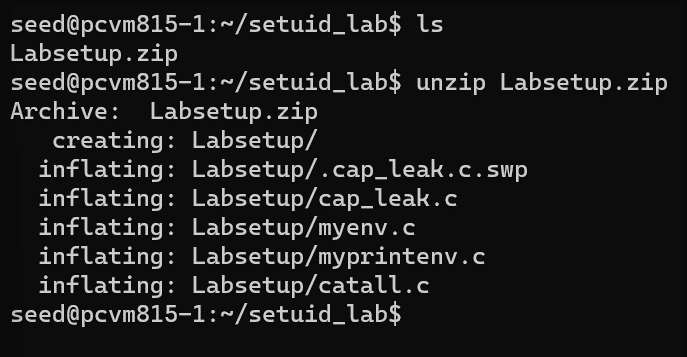
# Lab1: SEED Labs – Environment Variable and Set-UID Program

Mayur Suresh

CSC 302-01: Computer Security

Date: February 12th, 2023

## Task 1: Manipulating Environment Variables



The above picture shows the files under the Labsetup directory downloaded from the link ‘https://seedsecuritylabs.org/Labs\_20.04/Files/Environment\_Variable\_and\_SetUID/Labsetup.zip’

Using the seed account we use the env and the printenc command to print out the environment variables. Below we see both the env and the printenv commands used both giving us the environment variables.

Text

Description automatically generatedText

Description automatically generated

We can see specific environment variables by specifically inputting the variable as shown bellow with SHELL and PWD.

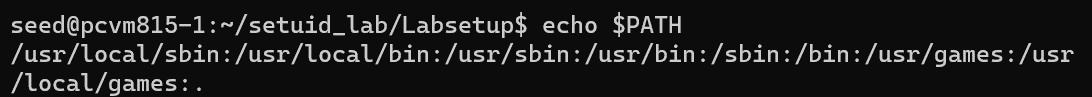
Text

Description automatically generated

Next we look at the file myenv.c with the `cat` command to check the file, then we check the PATH with the `echo $PATH` command. We see that there is still the path, which is still not reset and still showing the path address we used in the last lab.

Text

Description automatically generated



The next steps are to set up the countermeasures for the EV’s dynamic linking as discussed in class, which includes the steps below.

Text

Description automatically generated

Text

Description automatically generated

I run into the error shown below and this is because the LD\_PRELOAD still has the path from a previous exercise.

**Text

Description automatically generated**

Because LD\_PRELOAD was still pointing to the previous address we use the `export LD\_PRELOAD` command to set it to an empty string.

# Task 2: Passing Environment Variables from Parent Process to Child Process

For this task we will study how a child process gets its environment variables from its parent.

We first compile the myprintenv.c file found under the Labsetup directory previously downloaded. With this we get binary file a.out and we see below what is under the file after running it we see the output below.

Text

Description automatically generatedText

Description automatically generated

Next, we edit the myprintenv.c file and comment out the printenv() function under the child process and uncomment the function under the parent process. We then compile it again and we see the binary file reproduced and shown below.

Text

Description automatically generated Text

Description automatically generated

Finally, we compare the two environment variables and we see that there is no difference between them. This is because of the fork() that creates a child process which is essentially a copy of the parent process which inherits all the environment variables of the parent process. This makes them essentially the same thing at this creation. The newly created child process has a default of 0 which the why the switch statement works even without the printenv function that was commented out.

# Task 3: Environment Variables and execve()

For this task we will see how the environment variables are affected when a new program is executed with the execve(). This function calls a system call to load new commands executes it.

First we compile and run the myenv.c program with the command `gcc myenv.c`. This program executes a program called `/usr/bin/env/`.

Text

Description automatically generated 

After running and compiling the c file we get a `Permission denied’ message. This might be due to the NULL parameter for the execve function. We move forward to changing the NULL parameter under the execve function to `environ` from the \*\*environ variable. We go head and run it.

Text

Description automatically generatedText

Description automatically generated

Finally, we see the environment variables with the execve() function. After exchanging the NULL with the `environ` variable we are now able to access the environment variables.